Building a Simple Information Retrieval System using BM25 and GPT-3 and evaluated in the CISI collection.

Martin Emil Erismann Teschke

Exercício para seleção aluno especial - Curso "Deep Learning aplicado a sistemas de buscas", FEEC-Unicamp, primeiro semestre de 2023.

Este trabalho teve como objetivo explorar brevemente os tópicos: *Information Retrieval* (IR), BM25 *ranking algorithm*, com apoio do chatGPT utilizando o *Google Colab*.

Information Retrieval Systems são sistemas designados para auxiliar usuários na busca de informação em grandes coleções de documentos. A funcionalidade principal do sistema é retornar o(s) documento(s) que tenham a maior correspondência com a query de pesquisa do usuário. O processo de recuperação de informação possui várias etapas, entre elas o tratamento da query, busca no banco de dados e retorno dos itens relevantes de acordo com a query. Existem várias técnicas de indexação com *keyword indexing* e *semantic indexing*.

O algoritmo BM25 utiliza a indexação com base nas palavras por si só, deixando de lado a semântica, a qual é abordada em técnicas mais avançadas como SBERT. BM25 (Best Matching 25) funciona da seguinte forma:

1. Cada documento a ser ranqueado é representado como um vetor de termos, na qual cada termo é uma palavra ou expressão contida no documento.

2. O algoritmo avalia a frequência de cada termo no documento e calcula um peso para cada termo. O peso é calculado de acordo com a fórmula:

   w(i, D) = IDF(i) * ((k1 + 1) * f(i, D)) / (k1 * ((1 - b) + b * (|D| / avgdl)) + f(i, D))

   No qual:

   - w(i, D) é o peso do termo i no documento D
   - f(i, D) é a frequência do termo i no documento D
   - |D| é o comprimento do documento D (em número de termos)
   - avgdl é o comprimento médio dos documentos na coleção
   - k1 e b são constantes ajustáveis que afetam o cálculo do peso
   - IDF(i) é o inverso da frequência do termo i na coleção, dado por: IDF(i) = log((N - n(i) + 0.5) / (n(i) + 0.5)), onde N é o número total de documentos na coleção e n(i) é o número de documentos que contêm o termo i.

3. Para cada termo na consulta de busca, o algoritmo calcula um peso semelhante ao do documento, mas com algumas modificações para levar em conta a frequência do termo na consulta.

4. O algoritmo combina os pesos dos termos da consulta com os pesos dos termos do documento para obter uma pontuação de relevância para cada documento em relação à consulta. A pontuação é dada por:

   score(D, Q) = ∑ w(i, D) * w(i, Q)

   onde ∑ é a soma sobre todos os termos i na consulta Q.

5. Os documentos são classificados em ordem decrescente de pontuação de relevância e apresentados ao usuário como resultados da busca.

O algoritmo BM25 é amplamente utilizado em sistemas de busca e é conhecido por produzir resultados de alta qualidade. As constantes k1 e b podem ser ajustadas para otimizar o desempenho do algoritmo para uma determinada coleção de documentos e conjunto de consultas.

Os dados utilizados no trabalho foram obtidos do repositório da CISI *collection*, disponível em: http://ir.dcs.gla.ac.uk/resources/test_collections/cisi/ . Nas primeiras cinco linhas de códigos foi realizado o download dos dados e uma simples exploração. Tais linhas de códigos tiveram como apoio dois estudos apresentados no Kaggle: Some changes in CISI_EDA e CISI_EDA. Na sequência foi realizado o *import* das libs utilizadas para este projeto, sendo BM25Okapi e nltk as mais relevantes.

A classe BM25Okapi trás a implementação do algoritmo BM25 explicado anteriormente, por meio das funções get_top_n e get_scores. Sendo que a diferença principal é que a primeira retorna apenas uma quantidade n de documentos similares, enquanto a segunda retorna o score da query para a cada corpus da base. A biblioteca também possui uma função para transformar o corpus tanto da base quanto da query em tokens, porém a mesma não foi utilizada. Nesta etapa foi criado uma função a parte data_prep, pois assim além de realizar a tokenização foram retiradas as letras maiúsculas, pontuação, *stop words* (palavras comuns que aparecem em grande quantidade sem significado importante para a análise de texto). Tal preparação de dados melhora o desempenho do modelo. Na definição das palavras que estariam na lista de *stop words* foi utilizado a lib nltk.

Conclui-se que apesar de simples implementação a biblioteca BM25 atende com qualidade e agilidade a recuperação de informação.